

SCANF

Very susceptible to buffer overflow

Sean Barnum, Cigital, Inc. [vita¹]

Copyright © 2007 Cigital, Inc.

2007-04-04

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 8312 bytes

Attack Category	<ul style="list-style-type: none">• Malicious Input	
Vulnerability Category	<ul style="list-style-type: none">• Buffer Overflow• Format string• Unconditional	
Software Context	<ul style="list-style-type: none">• String Parsing	
Location	<ul style="list-style-type: none">• <code>stdio.h</code>	
Description	<p>The scanf family of functions scans input according to a format as described below. This format may contain conversion specifiers; the results from such conversions, if any, are stored through the pointer arguments. The scanf function reads input from the standard input stream stdin, fscanf reads input from the stream pointer stream, and sscanf reads its input from the character string pointed to by str.</p> <p>The vulnerability of the scanf() function resides in the fact that it has no bounds checking capability. If the string that is being accepted is longer than the buffer size, the characters will overflow into the adjoining memory space. This is a classic buffer overflow security vulnerability problem.</p> <p>The scanf() function is susceptible to buffer overflow.</p>	
APIs	Function Name	Comments
	<code>_cscanf</code>	fmt: 0; dst: 1 variable; Windows
	<code>_ftscanf</code>	fmt: 1; dst: 2 variable; Windows
	<code>_stscanf</code>	fmt: 1; dst: 2 variable; Windows
	<code>_tscanf</code>	fmt: 0; dst: 1 variable; Windows
	<code>fscanf</code>	fmt: 1; dst: 2 variable;

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

	ftscanf	fmt: 1; dst: 2 variable; Windows	
	fwscanf	fmt: 1; dst: 2 variable; Windows	
	scanf	fmt: 0; dst: 1 variable;	
	sscanf	fmt: 1; dst: 2 variable;	
	swscanf	fmt: 1; dst: 2 variable; Windows	
	vfscanf	fmt: 1; dst: 2 variable;	
	vftscanf	fmt: 1; dst: 2 variable; Windows	
	vscanf	fmt: 0; dst: 1 variable;	
	vsscanf	fmt: 1; dst: 2 variable;	
	wscanf	fmt: 0; dst: 1 variable; Windows	
Method of Attack		Attacker can overflow destination buffers of scanf() family with large input. Any "%s" in the format string leaves potential for this.	
Exception Criteria			
Solutions	Solution Applicability	Solution Description	Solution Efficacy
	All calls to scanf()	If you use the %s and %[conversions improperly, the number of characters read is limited only by where the next whitespace character appears. This almost certainly means that invalid input could make your program crash, because input too long would overflow whatever buffer you have provided for it. No matter how long your buffer is, a user could always supply	Effective

	<p>input that is longer.</p> <p>Fortunately, it is possible to avoid scanf buffer overflow by either specifying a field width or using a flag.</p> <p>When you specify a field width, you need to provide a buffer (using malloc or a similar function) of type char *. (See Memory Allocation for more information on malloc.) You need to make sure that the field width you specify does not exceed the number of bytes allocated to your buffer.</p>	
	<p>In GNU environments</p>	<p>On the other hand, you do not need to allocate a buffer if you specify the flag character --; scanf will do it for you. Simply pass scanf a pointer to an unallocated variable of type char *, and scanf will allocate however large a buffer the string requires and return the</p> <p>Effective</p>

	<div>result in your argument. This is a GNU-only extension to scanf functionality.</div>
Signature Details	<pre>#include <stdio.h> int scanf(const char *format, ...); int fscanf(FILE *stream, const char *format, ...); int sscanf(const char *str, const char *format, ...); #include <stdarg.h> int vscanf(const char *format, va_list ap); int vsscanf(const char *str, const char *format, va_list ap); int vfscanf(FILE *stream, const char *format, va_list ap);</pre>
Examples of Incorrect Code	<pre>int main() { char buff[15]={0}; printf("Enter your name:"); scanf(buff,"%s"); }</pre> <p>In this example, the program reads a string from the standard input but does not check the string's length. If the string has more than 14 characters, it causes a buffer overflow as scanf() tries to write the remaining character past buff's end.</p>
Examples of Corrected Code	<p>Here is a code example that shows first how to safely read a string of fixed maximum length by allocating a buffer and specifying a field width, then how to safely read a string of any length by using the flag.</p> <pre>#include <stdio.h> int main() { int bytes_read; int nbytes = 100; char *string1, *string2; string1 = (char *) malloc (25); puts ("Please enter a string of 20 characters or fewer."); scanf ("%20s", string1); printf ("\nYou typed the following string:\n%s\n\n", string1); puts ("Now enter a string of any length."); scanf ("%as", &string2);</pre>

	<pre>printf ("\nYou typed the following string:\n%s\n", string2); return 0; }</pre>	
Source References	<ul style="list-style-type: none"> • Viega, John & McGraw, Gary. <i>Building Secure Software: How to Avoid Security Problems the Right Way</i>. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, p. 146. • UNIX man page for scanf() • The GNU C programming tutorial² 	
Recommended Resource		
Discriminant Set	Operating Systems	<ul style="list-style-type: none"> • Windows • UNIX
	Languages	<ul style="list-style-type: none"> • C • C++

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com¹.

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. <mailto:copyright@cigital.com>